

Отчёт по лабораторной работе № 5

Дисциплина: Архитектура компьютера

Румянцев Артём Олегович

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
4	Выполнение лабораторной работы	9
4.1	Основы работы с тс	9
4.2	Структура программы на языке ассемблера NASM	12
4.3	Подключение внешнего файла	15
4.4	Выполнение заданий для самостоятельной работы	23
5	Выводы	31
6	Список литературы	32

Список иллюстраций

4.1	Открытый тс	9
4.2	Создание каталога	10
4.3	Перемещение между директориями	11
4.4	Создание файла	12
4.5	Открытие файла для редактирования	13
4.6	Открытие файла для просмотра	14
4.7	Компиляция файла и передача на обработку компоновщику	15
4.8	Исполнение файла	15
4.9	Скачанный файл	16
4.10	Копирование файла	17
4.11	Копирование файла	18
4.12	Редактирование файла	19
4.13	Исполнение файла	20
4.14	Отредактированный файл	21
4.15	Исполнение файла	22
4.16	Исполнение файла	23
4.17	Копирование файла	24
4.18	Редактирование файла	25
4.19	Исполнение файла	25
4.20	Копирование файла	27
4.21	Редактирование файла	28
4.22	Исполнение файла	28
4.23	Загрузка на сервер	30

Список таблиц

1 Цель работы

Целью данной лабораторной работы является приобретение практических навыков работы в Midnight Commander, освоение инструкций языка ассемблера `mov` и `int`.

2 Задание

1. Основы работы с тс
2. Структура программы на языке ассемблера NASM
3. Подключение внешнего файла
4. Выполнение заданий для самостоятельной работы

3 Теоретическое введение

Midnight Commander (или просто mc) — это программа, которая позволяет просматривать структуру каталогов и выполнять основные операции по управлению файловой системой, т.е. mc является файловым менеджером. Midnight Commander позволяет сделать работу с файлами более удобной и наглядной. Программа на языке ассемблера NASM, как правило, состоит из трёх секций: секция кода программы (SECTION .text), секция инициированных (известных во время компиляции) данных (SECTION .data) и секция неинициализированных данных (тех, под которые во время компиляции только отводится память, а значение присваивается в ходе выполнения программы) (SECTION .bss). Для объявления инициированных данных в секции .data используются директивы DB, DW, DD, DQ и DT, которые резервируют память и указывают, какие значения должны храниться в этой памяти: - DB (define byte) — определяет переменную размером в 1 байт; - DW (define word) — определяет переменную размером в 2 байта (слово); - DD (define double word) — определяет переменную размером в 4 байта (двойное слово); - DQ (define quad word) — определяет переменную размером в 8 байт (четырёх- рённое слово); - DT (define ten bytes) — определяет переменную размером в 10 байт. Директивы используются для объявления простых переменных и для объявления массивов. Для определения строк принято использовать директиву DB в связи с особенностями хранения данных в оперативной памяти. Инструкция языка ассемблера mov предназначена для дублирования данных источника в приёмнике.

```
mov dst,src
```

Здесь операнд `dst` — приёмник, а `src` — источник. В качестве операнда могут выступать регистры (`register`), ячейки памяти (`memory`) и непосредственные значения (`const`). Инструкция языка ассемблера `int` предназначена для вызова прерывания с указанным номером.

int n

Здесь `n` — номер прерывания, принадлежащий диапазону 0–255. При программировании в Linux с использованием вызовов ядра `sys_calls` `n=80h` (принято задавать в шестнадцатеричной системе счисления).

4 Выполнение лабораторной работы

4.1 Основы работы с mc

Открываю Midnight Commander, введя в терминал mc (рис. 01).

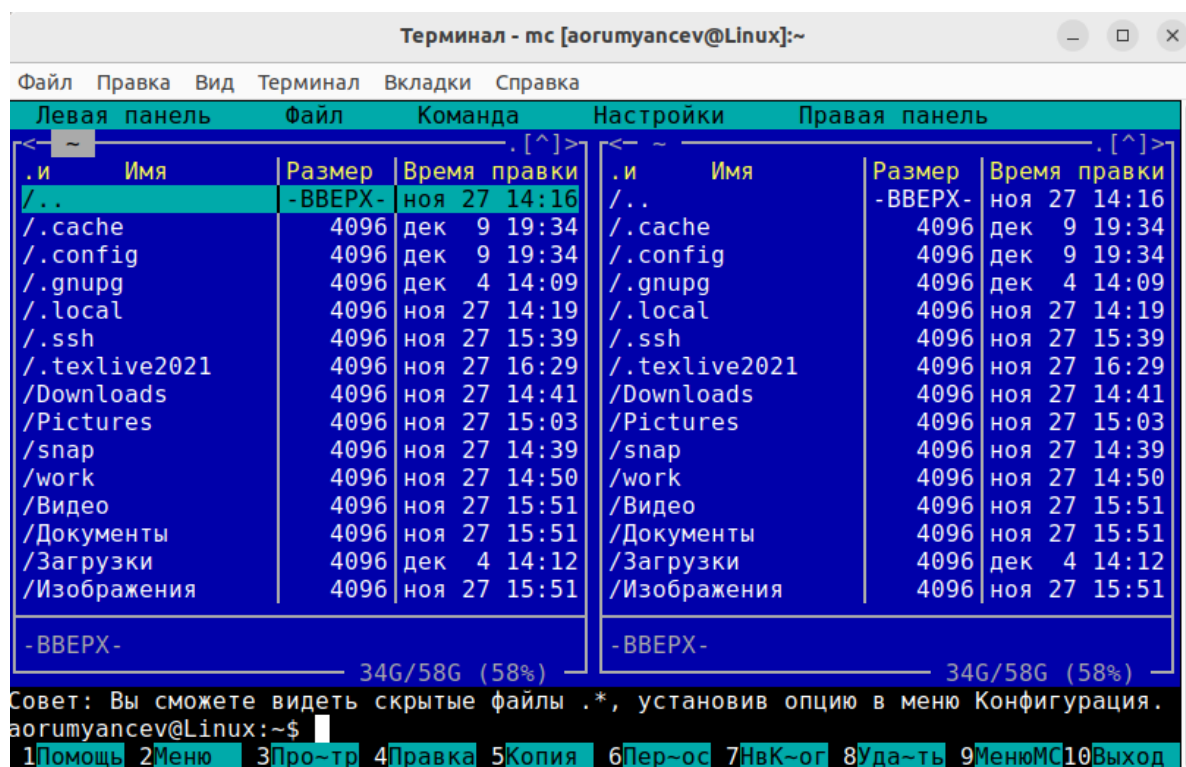


Рис. 4.1: Открытый mc

Перехожу в каталог ~/work/arch-рс, используя файловый менеджер mc

С помощью функциональной клавиши F7 создаю каталог lab05 (рис. 02).

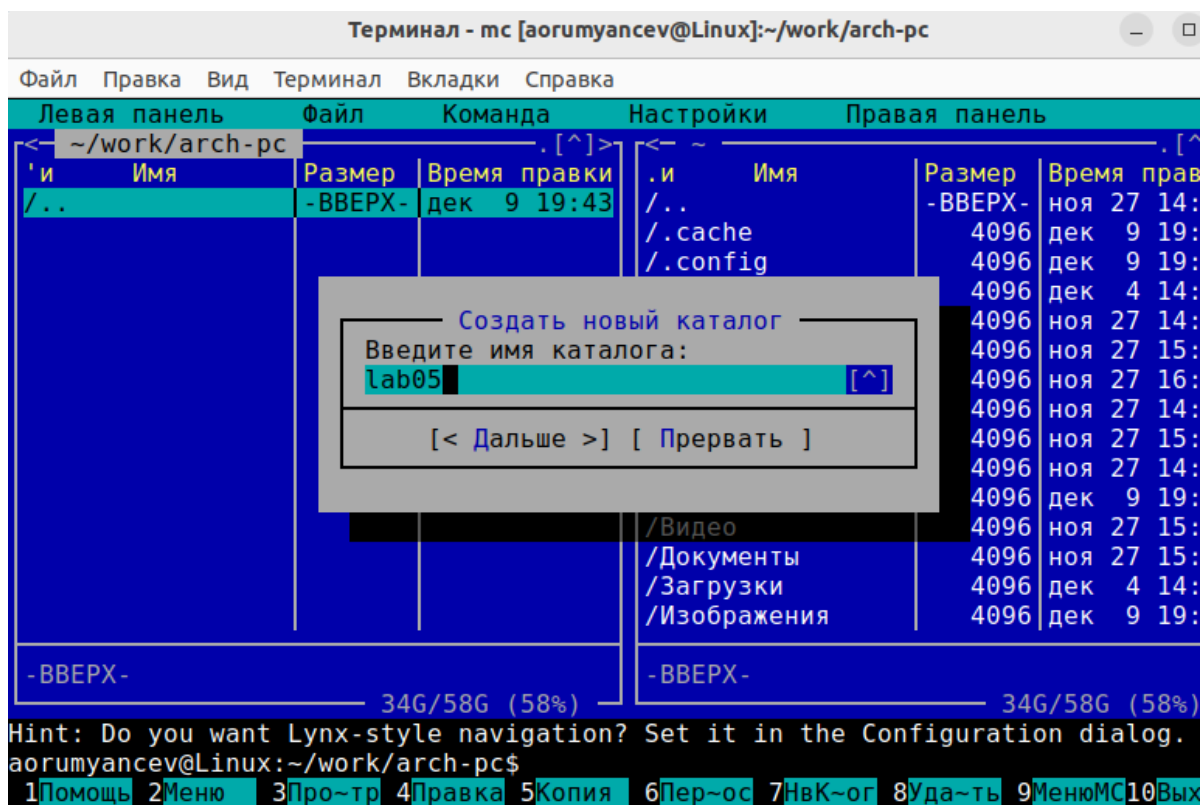


Рис. 4.2: Создание каталога

Переходу в созданный каталог (рис. 03).

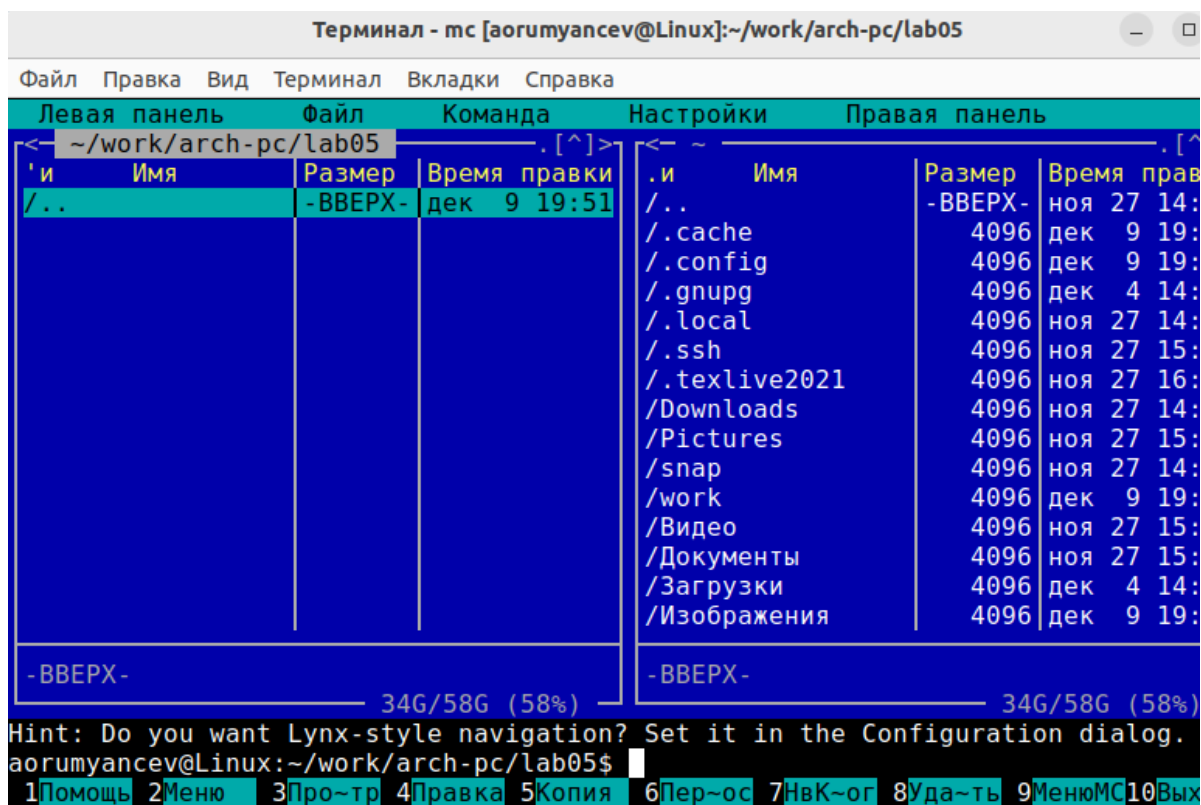


Рис. 4.3: Перемещение между директориями

В строке ввода прописываю команду `touch lab5-1.asm`, чтобы создать файл, в котором буду работать (рис. 04).

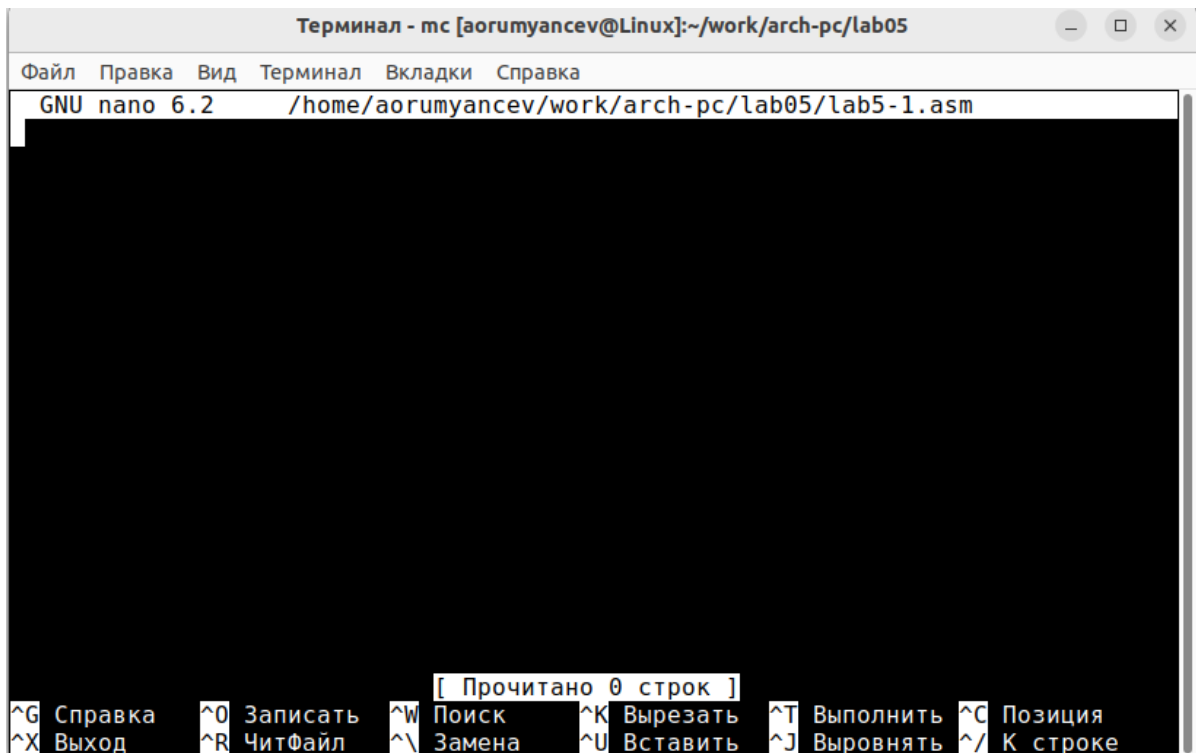


Рис. 4.5: Открытие файла для редактирования

Ввожу в файл код программы для запроса строки у пользователя. Далее выхожу из файла (Ctrl+X), сохраняя изменения (Y, Enter).

С помощью функциональной клавиши F3 открываю файл для просмотра, чтобы проверить, содержит ли файл текст программы (рис. 06).

```
Терминал - mc [aorumyancev@Linux]:~/work/arch-pc/lab05
Файл  Правка  Вид  Терминал  Вкладки  Справка
/home/aorumyancev/work/~ch-pc/lab05/lab5-1.asm  276/284  97%
SECTION .data
msg:  DB 'Введите строку:',10

msgLen:  EQU $-msg

SECTION .bss
buf1:    RESB 80
SECTION .text
GLOBAL _start
_start:
mov eax,4
mov ebx,1
mov ecx,msg
mov edx,msgLen
int 80h
mov eax, 3
mov ebx, 0
mov ecx, buf1
mov edx, 80
int 80h
mov eax,1
mov ebx,0
1Помощь 2Раз-рн 3Выход 4Нех 5Пер-ти 6 7Поиск 8Исх-ый 9Формат10Выход
```

Рис. 4.6: Открытие файла для просмотра

Транслирую текст программы файла в объектный файл командой `nasm -f elf lab5-1.asm`. Создался объектный файл `lab5-1.o`. Выполняю компоновку объектного файла с помощью команды `ld -m elf_i386 -o lab5-1 lab5-1.o` (рис. 7). Создался исполняемый файл `lab5-1`.

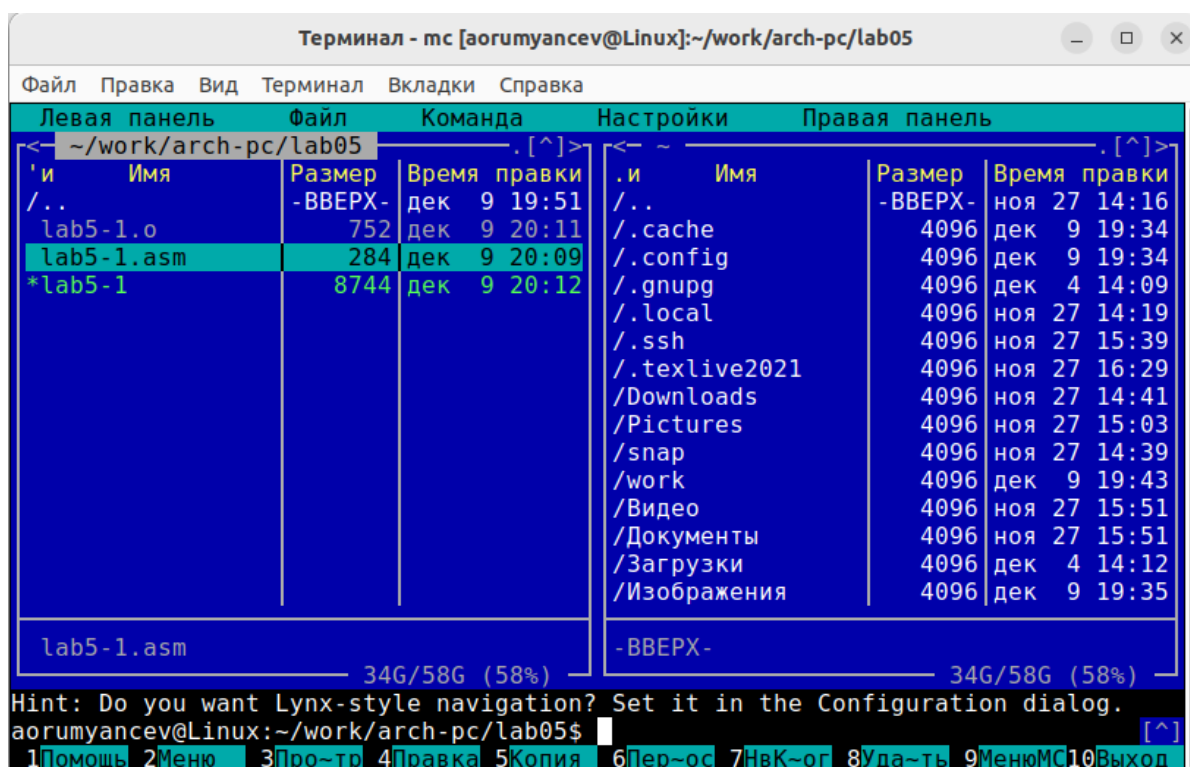


Рис. 4.7: Компиляция файла и передача на обработку компоновщику

Запускаю исполняемый файл. Программа выводит строку “Введите строку:” и ждет ввода с клавиатуры, я ввожу свои ФИО, на этом программа заканчивает свою работу (рис. 8).

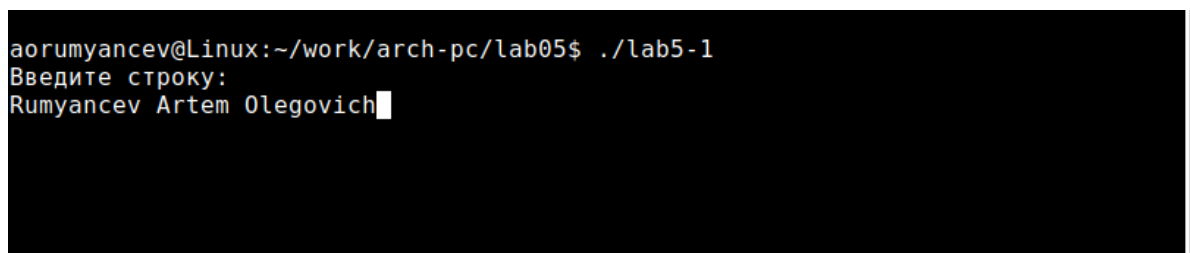


Рис. 4.8: Исполнение файла

4.3 Подключение внешнего файла

Скачиваю файл in_out.asm со страницы курса в ТУИС. Он сохранился в каталог “Загрузки” (рис. 9).

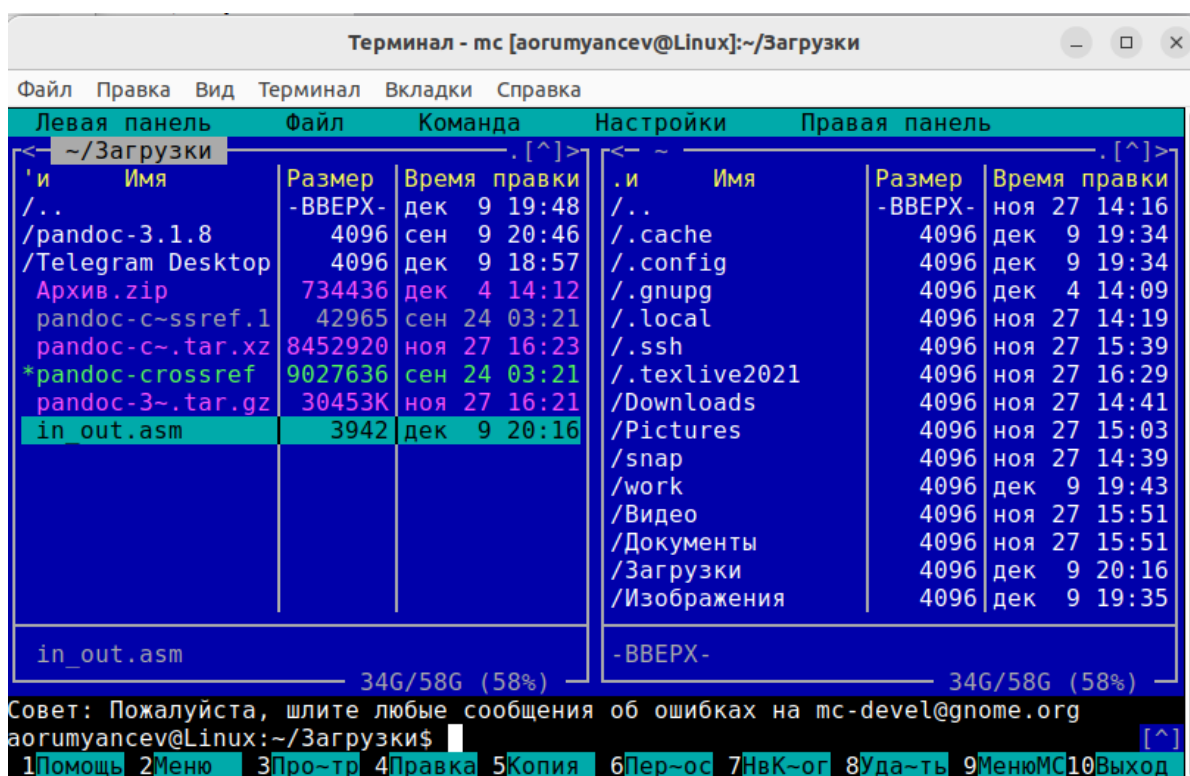


Рис. 4.9: Скачанный файл

С помощью функциональной клавиши F5 копирую файл in_out.asm из каталога Загрузки в созданный каталог lab05 (рис. 10).

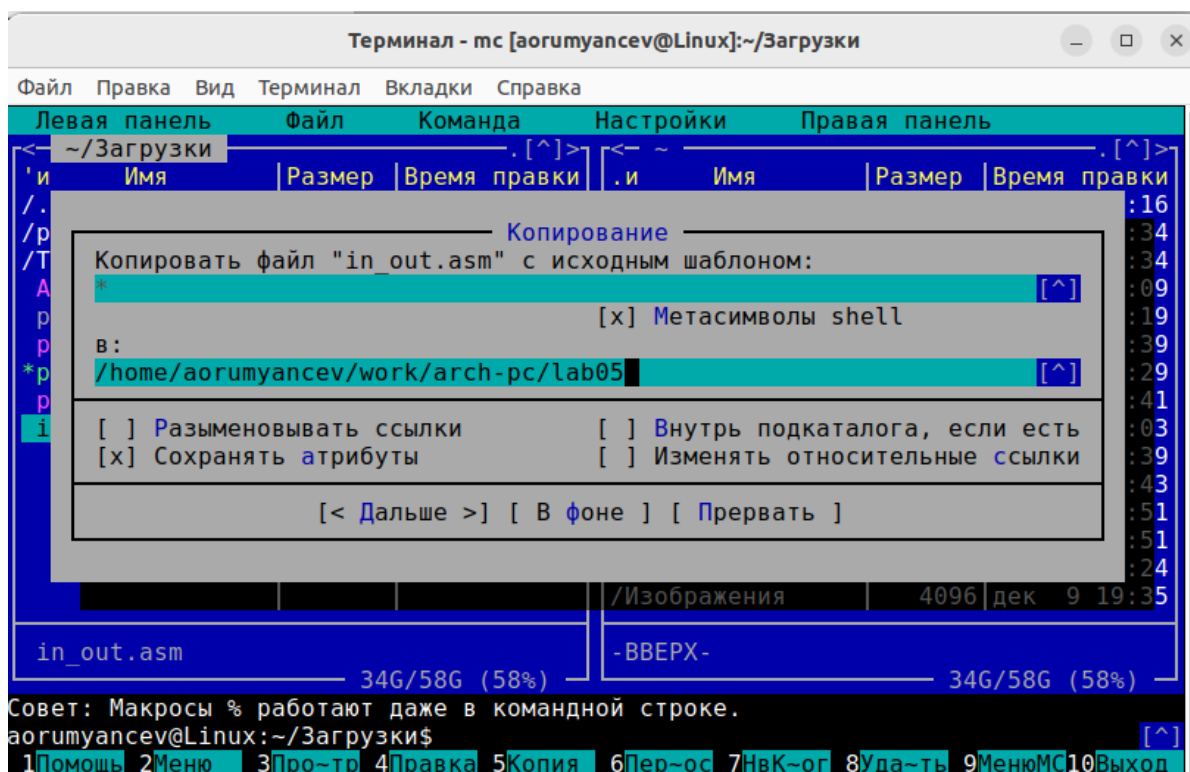


Рис. 4.10: Копирование файла

С помощью функциональной клавиши F5 копирую файл lab5-1 в тот же каталог, но с другим именем, для этого в появившемся окне ms прописываю имя для копии файла (рис. 11).

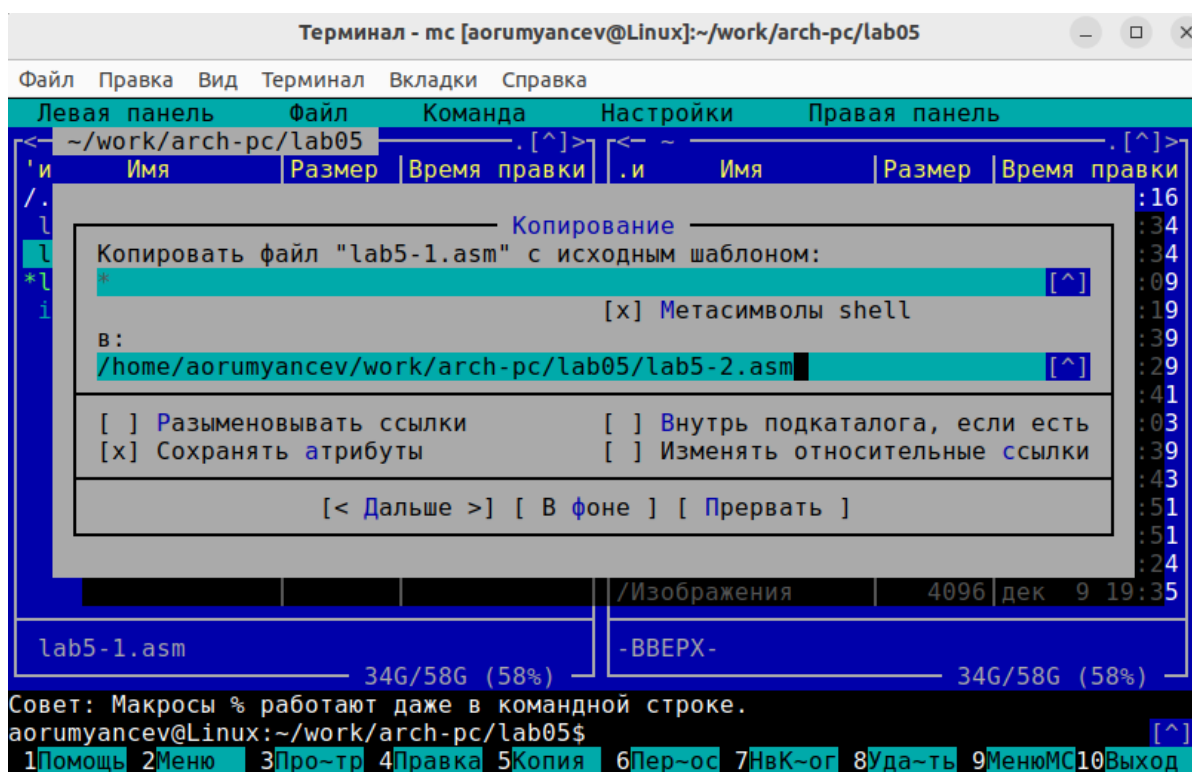
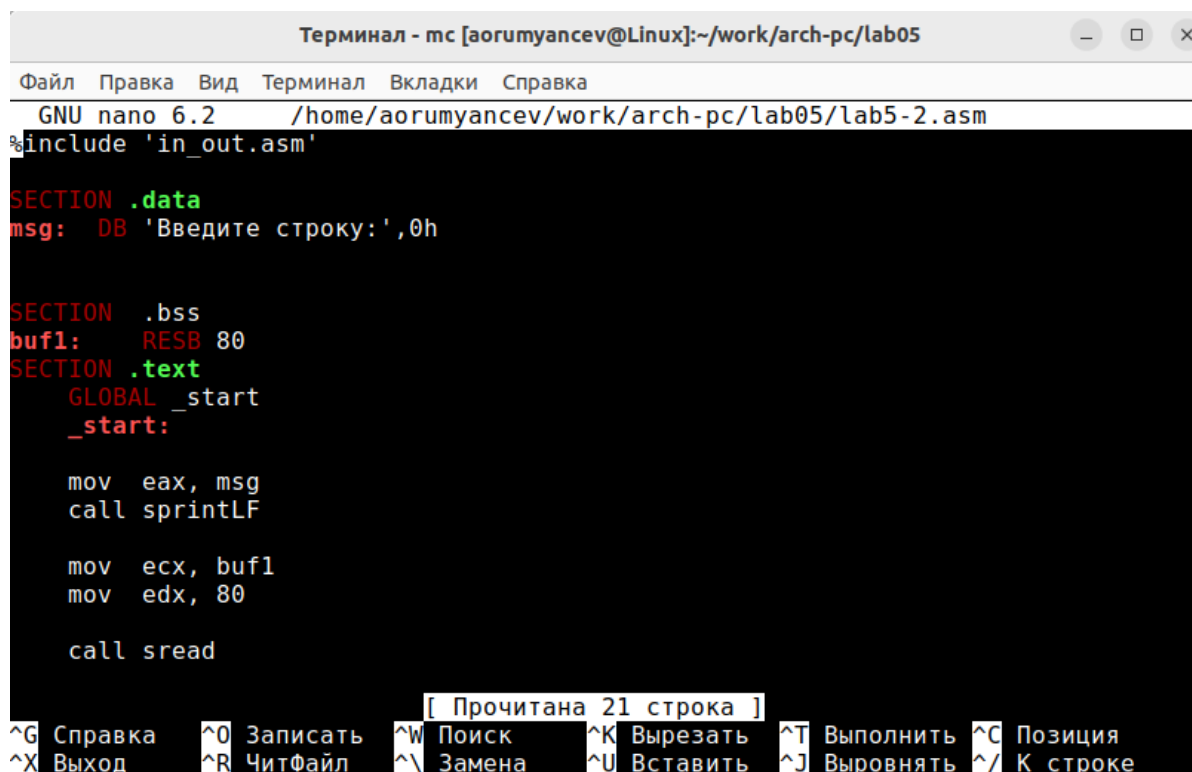


Рис. 4.11: Копирование файла

Изменяю содержимое файла lab5-2.asm во встроенном редакторе nano (рис. 12), чтобы в программе использовались подпрограммы из внешнего файла in_out.asm.



```
Терминал - mc [aorutyancev@Linux]:~/work/arch-pc/lab05
Файл  Правка  Вид  Терминал  Вкладки  Справка
GNU nano 6.2 /home/aorutyancev/work/arch-pc/lab05/lab5-2.asm
%include 'in_out.asm'

SECTION .data
msg: DB 'Введите строку:',0h

SECTION .bss
buf1: RESB 80
SECTION .text
GLOBAL _start
_start:

    mov     eax, msg
    call    sprintfLF

    mov     ecx, buf1
    mov     edx, 80

    call    sread

[ Прочитана 21 строка ]
^G Справка  ^O Записать  ^W Поиск    ^K Вырезать  ^T Выполнить  ^C Позиция
^X Выход    ^R ЧитФайл  ^\ Замена  ^U Вставить  ^J Выводить   ^/_ К строке
```

Рис. 4.12: Редактирование файла

Транслирую текст программы файла в объектный файл командой `nasm -f elf lab5-2.asm`. Создался объектный файл `lab5-2.o`. Выполняю компоновку объектного файла с помощью команды `ld -m elf_i386 -o lab5-2 lab5-2.o` Создался исполняемый файл `lab5-2`. Запускаю исполняемый файл (рис. 13).

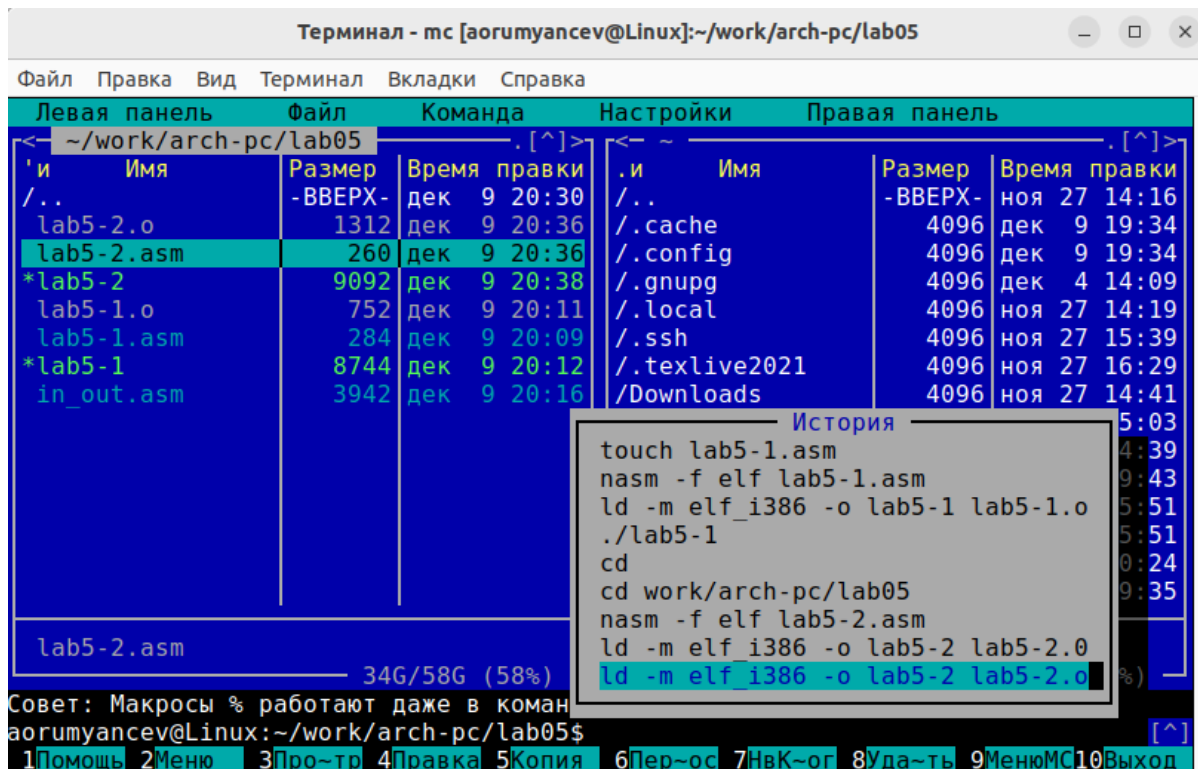
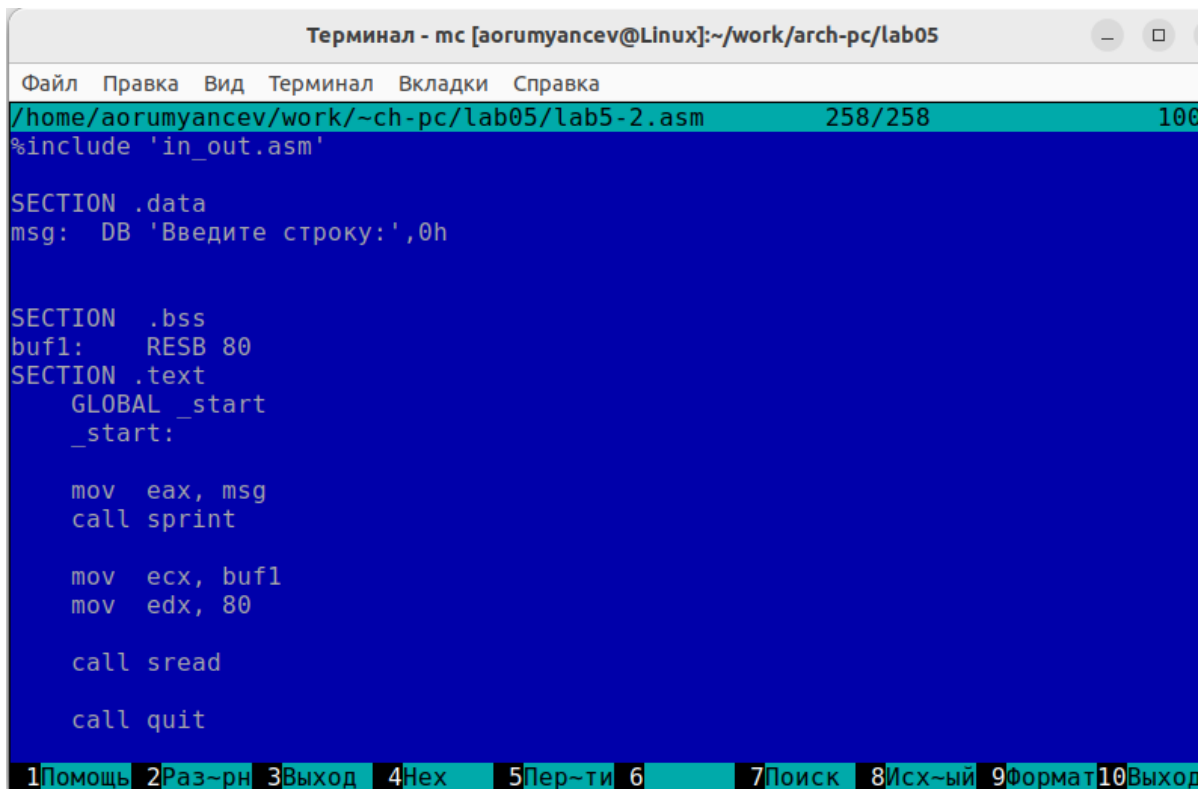


Рис. 4.13: Исполнение файла

Открываю файл lab5-2.asm для редактирования в nano функциональной клавишей F4. Изменяю в нем подпрограмму sprintLF на sprint. Сохраняю изменения и открываю файл для просмотра, чтобы проверить сохранение действий (рис. 14).



```
Терминал - mc [aorummyancev@Linux]:~/work/arch-pc/lab05
Файл  Правка  Вид  Терминал  Вкладки  Справка
/home/aorummyancev/work/~ch-pc/lab05/lab5-2.asm  258/258  100%
#include 'in_out.asm'

SECTION .data
msg: DB 'Введите строку:',0h

SECTION .bss
buf1: RESB 80
SECTION .text
GLOBAL _start
_start:

    mov  eax, msg
    call sprint

    mov  ecx, buf1
    mov  edx, 80

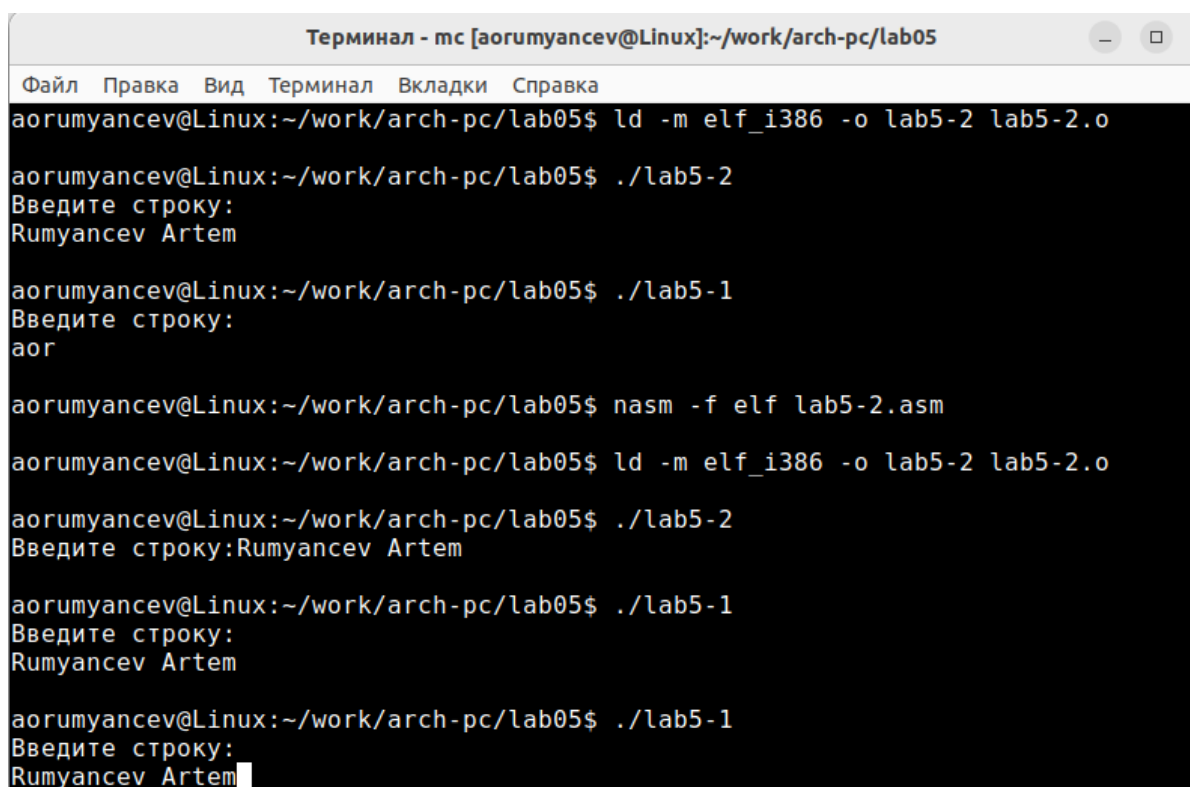
    call sread

    call quit
```

1Помощь 2Раз~рн 3Выход 4Нех 5Пер~ти 6 7Поиск 8Исх~ый 9Формат10Выход

Рис. 4.14: Отредактированный файл

Снова транслирую файл, выполняю компоновку созданного объектного файла, запускаю новый исполняемый файл (рис. 15,16).

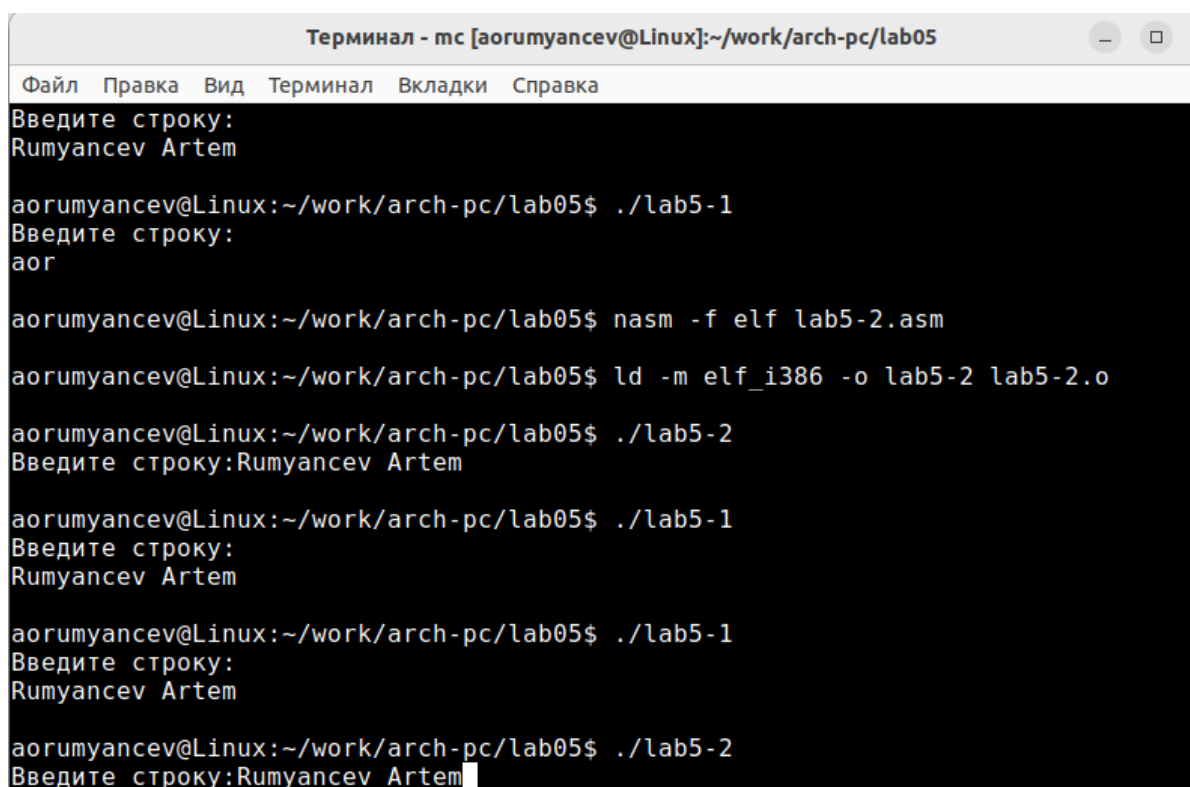


Терминал - mc [aorumyancev@Linux]:~/work/arch-pc/lab05

Файл Правка Вид Терминал Вкладки Справка

```
aorumyancev@Linux:~/work/arch-pc/lab05$ ld -m elf_i386 -o lab5-2 lab5-2.o
aorumyancev@Linux:~/work/arch-pc/lab05$ ./lab5-2
Введите строку:
Rumyancev Artem
aorumyancev@Linux:~/work/arch-pc/lab05$ ./lab5-1
Введите строку:
aor
aorumyancev@Linux:~/work/arch-pc/lab05$ nasm -f elf lab5-2.asm
aorumyancev@Linux:~/work/arch-pc/lab05$ ld -m elf_i386 -o lab5-2 lab5-2.o
aorumyancev@Linux:~/work/arch-pc/lab05$ ./lab5-2
Введите строку:Rumyancev Artem
aorumyancev@Linux:~/work/arch-pc/lab05$ ./lab5-1
Введите строку:
Rumyancev Artem
```

Рис. 4.15: Исполнение файла



```
Терминал - mc [aorumyancev@Linux]:~/work/arch-pc/lab05
Файл  Правка  Вид  Терминал  Вкладки  Справка
Введите строку:
Rumyancev Artem

aorumyancev@Linux:~/work/arch-pc/lab05$ ./lab5-1
Введите строку:
aor

aorumyancev@Linux:~/work/arch-pc/lab05$ nasm -f elf lab5-2.asm
aorumyancev@Linux:~/work/arch-pc/lab05$ ld -m elf_i386 -o lab5-2 lab5-2.o
aorumyancev@Linux:~/work/arch-pc/lab05$ ./lab5-2
Введите строку:Rumyancev Artem

aorumyancev@Linux:~/work/arch-pc/lab05$ ./lab5-1
Введите строку:
Rumyancev Artem

aorumyancev@Linux:~/work/arch-pc/lab05$ ./lab5-1
Введите строку:
Rumyancev Artem

aorumyancev@Linux:~/work/arch-pc/lab05$ ./lab5-2
Введите строку:Rumyancev Artem
```

Рис. 4.16: Исполнение файла

Разница между первым исполняемым файлом lab5-1 и вторым lab5-2 в том, что запуск первого запрашивает ввод с новой строки, а программа, которая выполняется при запуске второго, запрашивает ввод без переноса на новую строку, потому что в этом заключается различие между подпрограммами `sprintf` и `sprint`.

4.4 Выполнение заданий для самостоятельной работы

1. Создаю копию файла lab5-1.asm с именем lab5-1-1.asm с помощью функциональной клавиши F5 (рис. 17).

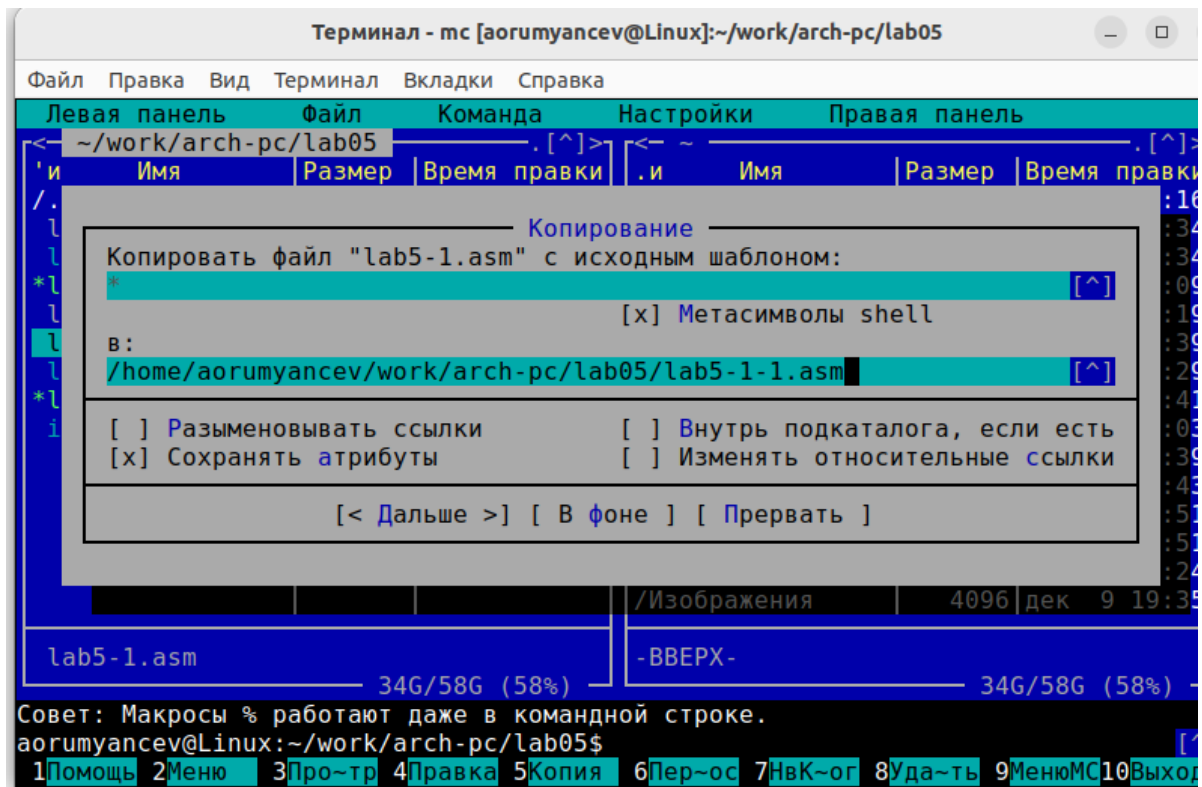
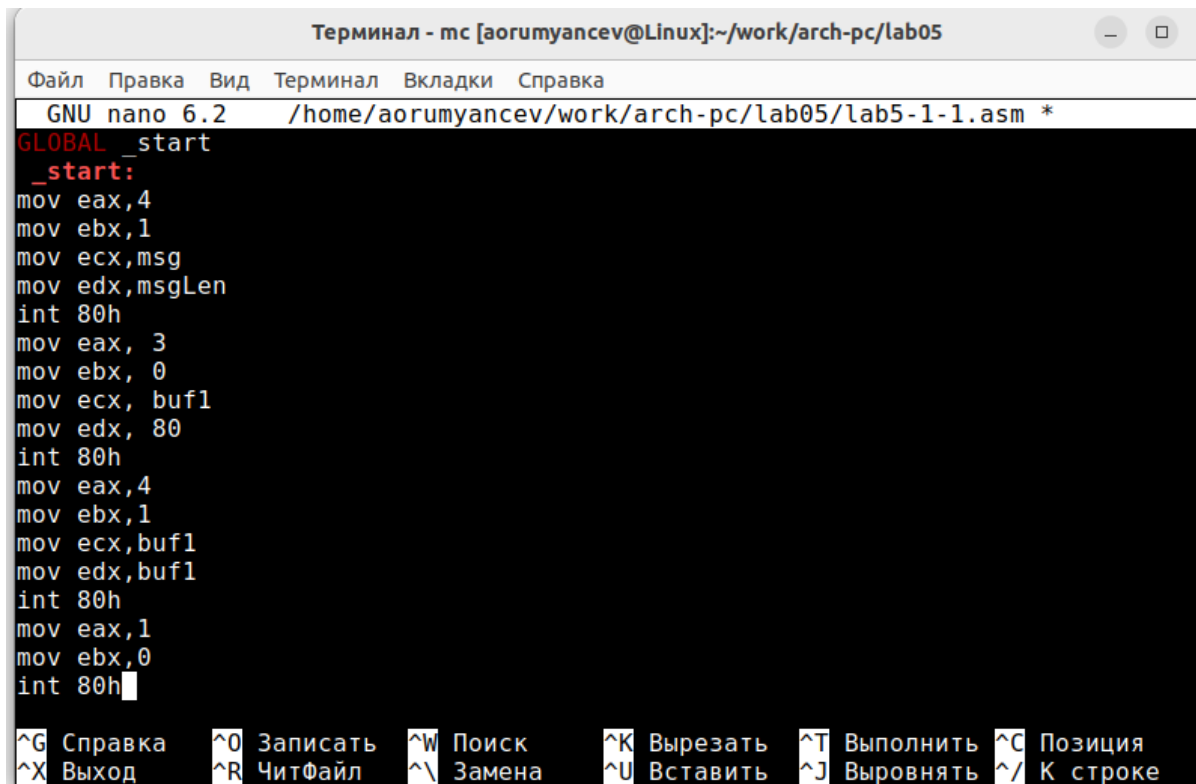


Рис. 4.17: Копирование файла

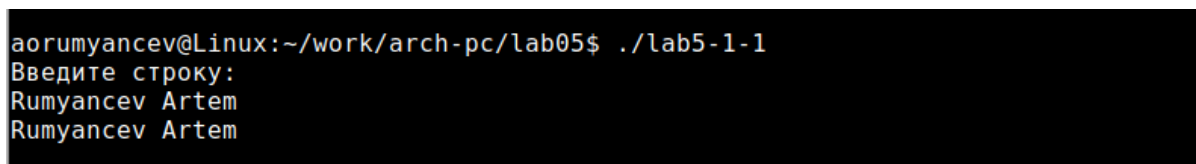
С помощью функциональной клавиши F4 открываю созданный файл для редактирования. Изменяю программу так, чтобы кроме вывода приглашения и запроса ввода, она выводила вводимую пользователем строку (рис. 18).



```
Терминал - мс [aorummyancev@Linux]:~/work/arch-pc/lab05
Файл  Правка  Вид  Терминал  Вкладки  Справка
GNU nano 6.2 /home/aorummyancev/work/arch-pc/lab05/lab5-1-1.asm *
GLOBAL _start
_start:
mov eax,4
mov ebx,1
mov ecx,msg
mov edx,msgLen
int 80h
mov eax, 3
mov ebx, 0
mov ecx, buf1
mov edx, 80
int 80h
mov eax,4
mov ebx,1
mov ecx,buf1
mov edx,buf1
int 80h
mov eax,1
mov ebx,0
int 80h
^G Справка  ^O Записать  ^W Поиск    ^K Вырезать  ^T Выполнить  ^C Позиция
^X Выход     ^R ЧитФайл  ^\ Замена   ^U Вставить  ^J Выводить   ^/ К строке
```

Рис. 4.18: Редактирование файла

2. Создаю объектный файл lab5-1-1.o, отдаю его на обработку компоновщику, получаю исполняемый файл lab5-1-1, запускаю полученный исполняемый файл. Программа запрашивает ввод, ввожу свои ФИО, далее программа выводит введенные мною данные (рис. 19).



```
aorummyancev@Linux:~/work/arch-pc/lab05$ ./lab5-1-1
Введите строку:
Rumyancev Artem
Rumyancev Artem
```

Рис. 4.19: Исполнение файла

Код программы из пункта 2:

```
SECTION .data ; Секция инициированных данных
msg: DB 'Введите строку:',10
```

```

msgLen: EQU $-msg ; Длина переменной 'msg'
SECTION .bss ; Секция не инициированных данных
buf1: RESB 80 ; Буфер размером 80 байт
SECTION .text ; Код программы
GLOBAL _start ; Начало программы
_start: ; Точка входа в программу
mov eax,4 ; Системный вызов для записи (sys_write)
mov ebx,1 ; Описатель файла 1 - стандартный вывод
mov ecx,msg ; Адрес строки 'msg' в 'ecx'
mov edx,msgLen ; Размер строки 'msg' в 'edx'
int 80h ; Вызов ядра
mov eax, 3 ; Системный вызов для чтения (sys_read)
mov ebx, 0 ; Дескриптор файла 0 - стандартный ввод
mov ecx, buf1 ; Адрес буфера под вводимую строку
mov edx, 80 ; Длина вводимой строки
int 80h ; Вызов ядра
mov eax,4 ; Системный вызов для записи (sys_write)
mov ebx,1 ; Описатель файла '1' - стандартный вывод
mov ecx,buf1 ; Адрес строки buf1 в ecx
mov edx,buf1 ; Размер строки buf1
int 80h ; Вызов ядра
mov eax,1 ; Системный вызов для выхода (sys_exit)
mov ebx,0 ; Выход с кодом возврата 0 (без ошибок)
int 80h ; Вызов ядра

```

3. Создаю копию файла lab5-2.asm с именем lab5-2-1.asm с помощью функциональной клавиши F5 (рис. 20).

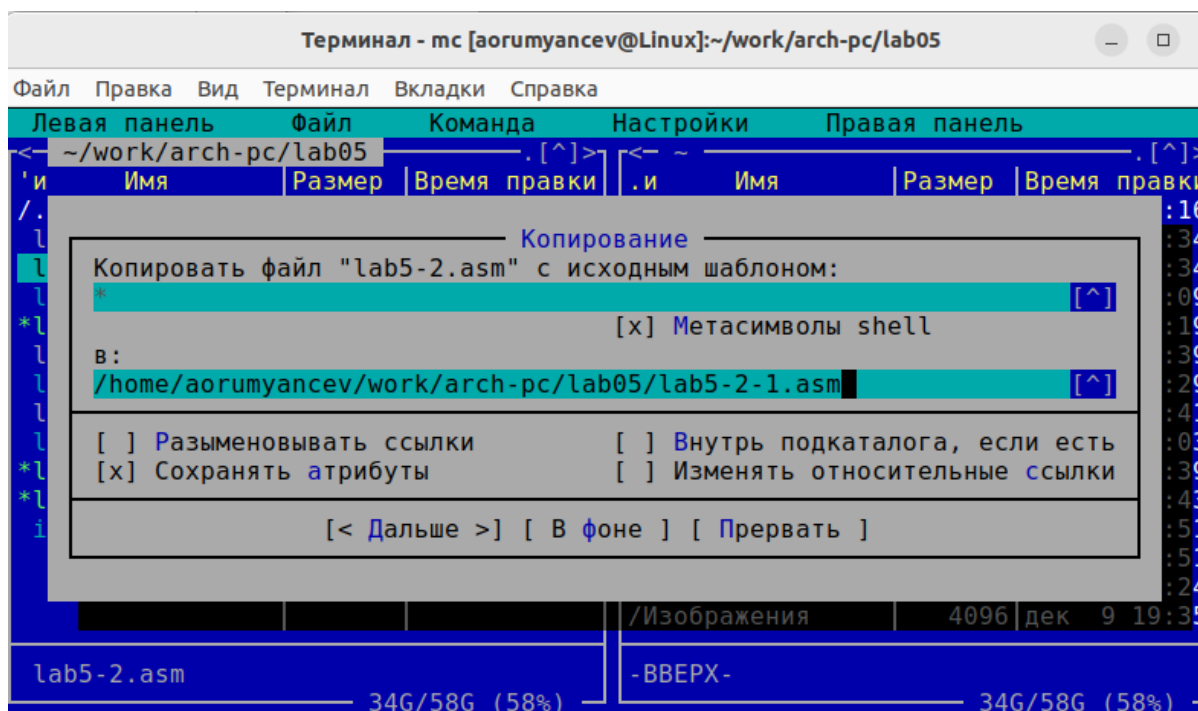
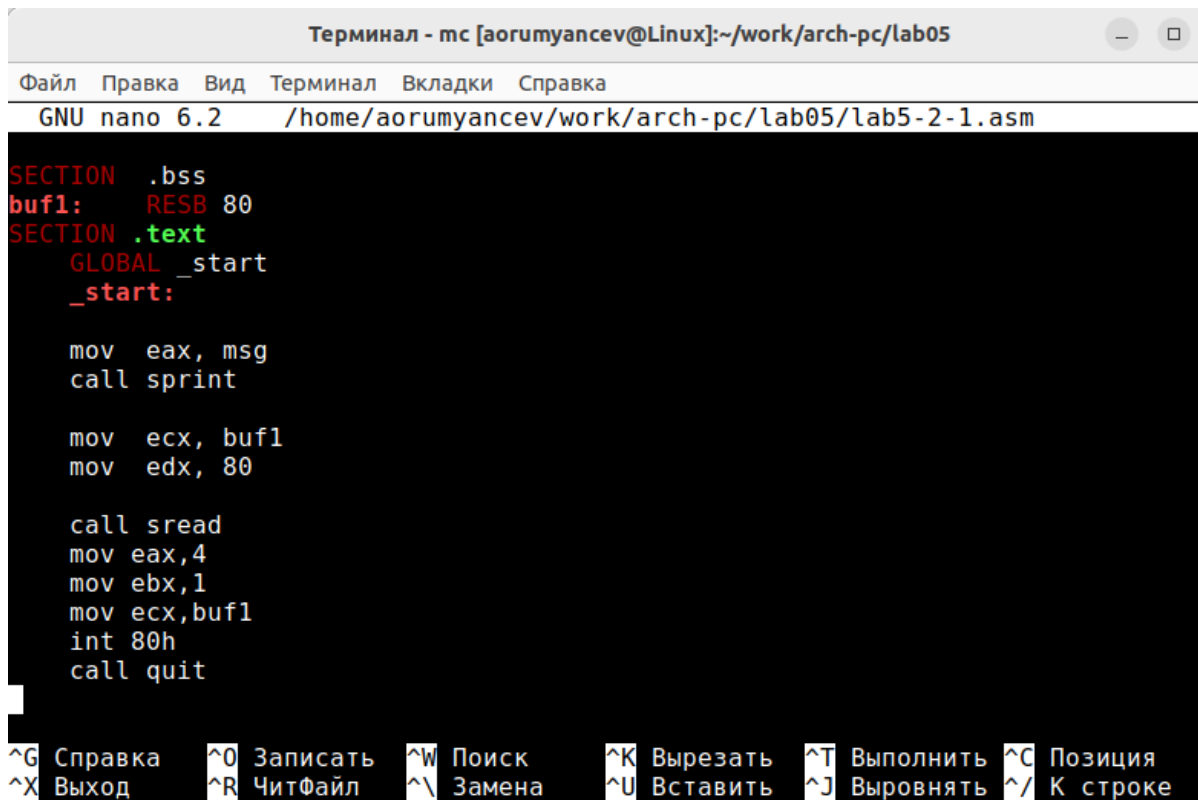


Рис. 4.20: Копирование файла

С помощью функциональной клавиши F4 открываю созданный файл для редактирования. Изменяю программу так, чтобы кроме вывода приглашения и запроса ввода, она выводила вводимую пользователем строку (рис. 21).



```
Терминал - mc [aorummyancev@Linux]:~/work/arch-pc/lab05
Файл  Правка  Вид  Терминал  Вкладки  Справка
GNU nano 6.2  /home/aorummyancev/work/arch-pc/lab05/lab5-2-1.asm

SECTION .bss
buf1:   RESB 80
SECTION .text
GLOBAL _start
_start:

    mov     eax, msg
    call    sprint

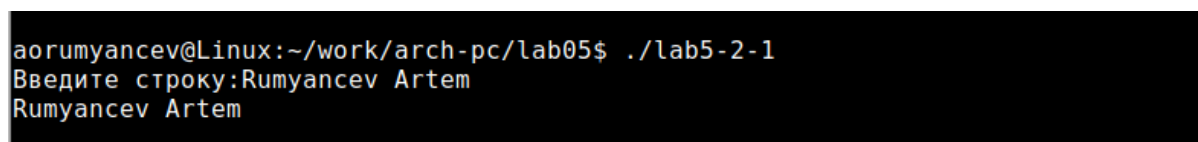
    mov     ecx, buf1
    mov     edx, 80

    call    sread
    mov     eax, 4
    mov     ebx, 1
    mov     ecx, buf1
    int     80h
    call    quit

^G Справка  ^O Записать  ^W Поиск    ^K Вырезать  ^T Выполнить  ^C Позиция
^X Выход    ^R ЧитФайл  ^\ Замена   ^U Вставить  ^J Выровнять  ^/_ К строке
```

Рис. 4.21: Редактирование файла

4. Создаю объектный файл lab5-2-1.o, отдаю его на обработку компоновщику, получаю исполняемый файл lab5-2-1, запускаю полученный исполняемый файл. Программа запрашивает ввод без переноса на новую строку, ввожу свои ФИО, далее программа выводит введенные мною данные (рис. 22).



```
aorummyancev@Linux:~/work/arch-pc/lab05$ ./lab5-2-1
Введите строку:Rummyancev Artem
Rummyancev Artem
```

Рис. 4.22: Исполнение файла

Код программы из пункта 3:

```
%include 'in_out.asm'

SECTION .data ; Секция инициированных данных
msg: DB 'Введите строку: ',0h ; сообщение
```

```

SECTION .bss ; Секция не инициированных данных
buf1: RESB 80 ; Буфер размером 80 байт
SECTION .text ; Код программы
GLOBAL _start ; Начало программы
_start: ; Точка входа в программу
mov eax, msg ; запись адреса выводимого сообщения в `EAX`
call sprint ; вызов подпрограммы печати сообщения
mov ecx, buf1 ; запись адреса переменной в `EAX`
mov edx, 80 ; запись длины вводимого сообщения в `EBX`
call sread ; вызов подпрограммы ввода сообщения
mov eax, 4 ; Системный вызов для записи (sys_write)
mov ebx, 1 ; Описатель файла '1' - стандартный вывод
mov ecx, buf1 ; Адрес строки buf1 в ecx
int 80h ; Вызов ядра
call quit ; вызов подпрограммы завершения

```

5. С помощью команд `git add .`, `git commit -m 'Add files'`, `git push` добавляю файлы лабораторной в репозиторий GitHub

```
Терминал - aorumyancev@Linux: ~/work/study/2023-2024/Computer Arch/arch-pc/labs/lab05/report
Файл Правка Вид Терминал Вкладки Справка
aorumyancev@Linux:~/work/study/2023-2024/Computer Arch/arch-pc/labs/lab05$ git add .
aorumyancev@Linux:~/work/study/2023-2024/Computer Arch/arch-pc/labs/lab05$ git commit -m "Добавил файлы"
[master 6592365] Добавил файлы
Committer: aorumyancev <aorumyancev@Linux.myquest.virtualbox.org>
Ваше имя или электронная почта настроены автоматически на основании вашего имени пользователя и имени машины. Пожалуйста, проверьте, что они определены правильно.
Вы можете отключить это уведомление установив их напрямую. Запустите следующую команду и следуйте инструкциям вашего текстового редактора, для редактирования вашего файла конфигурации:

git config --global --edit

После этого, изменить авторство этой коммита можно будет с помощью команды:

git commit --amend --reset-author

13 files changed, 266 insertions(+)
create mode 100644 labs/lab05/report/in_out.asm
create mode 100755 labs/lab05/report/lab5-1
create mode 100755 labs/lab05/report/lab5-1-1
create mode 100644 labs/lab05/report/lab5-1-1.asm
create mode 100644 labs/lab05/report/lab5-1-1.o
create mode 100644 labs/lab05/report/lab5-1.asm
create mode 100644 labs/lab05/report/lab5-1.o
create mode 100755 labs/lab05/report/lab5-2
create mode 100755 labs/lab05/report/lab5-2-1
create mode 100644 labs/lab05/report/lab5-2-1.asm
create mode 100644 labs/lab05/report/lab5-2-1.o
create mode 100644 labs/lab05/report/lab5-2.asm
create mode 100644 labs/lab05/report/lab5-2.o
aorumyancev@Linux:~/work/study/2023-2024/Computer Arch/arch-pc/labs/lab05$ git push
Перечисление объектов: 22, готово.
Подсчет объектов: 100% (22/22), готово.
При сжатии изменений используется до 6 потоков
Сжатие объектов: 100% (18/18), готово.
Запись объектов: 100% (18/18), 4.74 КиБ | 4.74 МБ/с, готово.
Всего 18 (изменений 10), повторно использовано 0 (изменений 0), повторно использовано пакетов 0
remote: Resolving deltas: 100% (10/10), completed with 2 local objects.
To github.com:aorumyancev/study_2023-2024_arch-pc.git
1f3de8e..6592365 master -> master
```

Рис. 4.23: Загрузка на сервер

5 Выводы

При выполнении данной лабораторной работы я приобрел практические навыки работы в Midnight Commander, а также освоил инструкции языка ассемблера `mov` и `int`.

6 Список литературы

1. Лабораторная работа №5